PCT WORLD INTELLECTUAL PROPERTY ORGANIZATION
International Bureau

INTERNATIONAL APPLICATION PUBLISHED UNDER THE PATENT COOPERATION TREATY (PCT)

(51) International Patent Classification 7 : G06F 12/02 — A1

(11) International Publication Number: WO 00/58838

(43) International Publication Date: 5 October 2000 (05.10.00)

(21) International Application Number: PCT/GB00/01063

(22) International Filing Date: 21 March 2000 (21.03.00)

(30) Priority Data:
2,267,484 30 March 1999 (30.03.99) CA

(71) Applicant: INTERNATIONAL BUSINESS MACHINES CORPORATION [US/US]; New Orchard Road, Armonk, New York, NY 10504 (US).

(71) Applicant *(for MC only)*: IBM UNITED KINGDOM LIMITED [GB/GB]; North Harbour, P.O. Box 41, Portsmouth, Hampshire PO6 3AU (GB).

(72) Inventors: BITNER, Deloy, Pehrson; 5278 West Melinda Lane, Glendale, AZ 85308 (US). CLOHESSY, Kim; 14625 North 62nd Way, Scottsdale, AZ 85254 (US). ORN, Mikael; 4001 North 54th Place, Phoenix, AZ 85308 (US).

(74) Agent: WALDNER, Philip; IBM United Kingdom Limited, Intellectual Property Law, Hursley Park, Winchester, Hampshire SO21 2JN (GB).

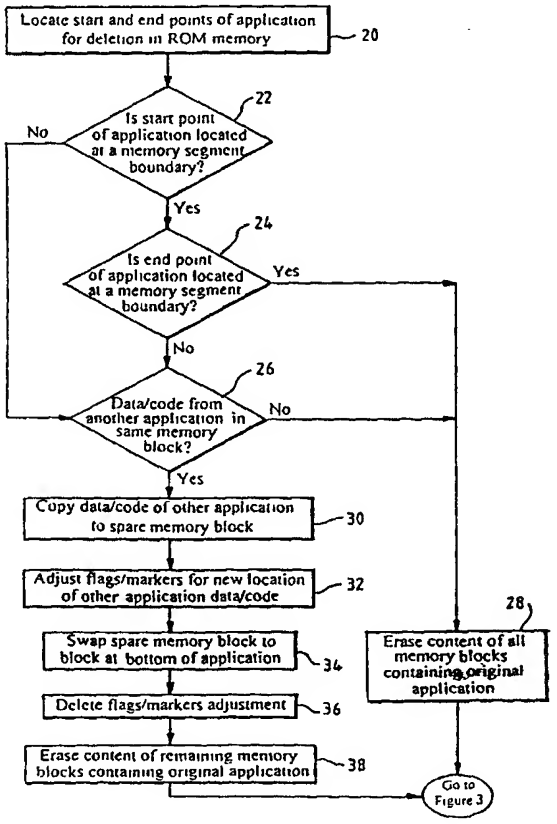
(81) Designated States: AE, AL, AM, AT, AU, AZ, BA, BB, BG, BR, BY, CH, CN, CR, CU, CZ, DE, DK, DM, EE, ES, FI, GB, GD, GE, GH, GM, HR, HU, ID, IL, IN, IS, JP, KE, KG, KP, KR, KZ, LC, LK, LR, LS, LT, LU, LV, MA, MD, MG, MK, MN, MW, MX, NO, NZ, PL, PT, RO, RU, SD, SE, SG, SI, SK, SL, TJ, TM, TR, TT, TZ, UA, UG, UZ, VN, YU, ZA, ZW, ARIPO patent (GH, GM, KE, LS, MW, SD, SL, SZ, TZ, UG, ZW), Eurasian patent (AM, AZ, BY, KG, KZ, MD, RU, TJ, TM), European patent (AT, BE, CH, CY, DE, DK, ES, FI, FR, GB, GR, IE, IT, LU, MC, NL, PT, SE), OAPI patent (BF, BJ, CF, CG, CI, CM, GA, GN, GW, ML, MR, NE, SN, TD, TG).

Published
With international search report.

(54) Title: RECLAIMING MEMORY FROM DELETED APPLICATIONS

(57) Abstract

The invention provides a method for removing code (applications and data) from read–only memory, and compacting the remaining code in memory either as an application is deleted or when there is not sufficient room to hold a new application. One or more "spare" memory segments are reserved for use during compaction. Where the code for removal shares a memory segment with other code that is not be removed, the other code is copied to a spare memory segment, and then swapped back to its original location. The code can then be compacted to remove the "holes" left by the erased code.



FOR THE PURPOSES OF INFORMATION ONLY

Codes used to identify States party to the PCT on the front pages of pamphlets publishing international applications under the PCT.

AL	Albania	**ES**	Spain	**LS**	Lesotho	**SI**	Slovenia
AM	Armenia	**FI**	Finland	**LT**	Lithuania	**SK**	Slovakia
AT	Austria	**FR**	France	**LU**	Luxembourg	**SN**	Senegal
AU	Australia	**GA**	Gabon	**LV**	Latvia	**SZ**	Swaziland
AZ	Azerbaijan	**GB**	United Kingdom	**MC**	Monaco	**TD**	Chad
BA	Bosnia and Herzegovina	**GE**	Georgia	**MD**	Republic of Moldova	**TG**	Togo
BB	Barbados	**GH**	Ghana	**MG**	Madagascar	**TJ**	Tajikistan
BE	Belgium	**GN**	Guinea	**MK**	The former Yugoslav Republic of Macedonia	**TM**	Turkmenistan
BF	Burkina Faso	**GR**	Greece			**TR**	Turkey
BG	Bulgaria	**HU**	Hungary	**ML**	Mali	**TT**	Trinidad and Tobago
BJ	Benin	**IE**	Ireland	**MN**	Mongolia	**UA**	Ukraine
BR	Brazil	**IL**	Israel	**MR**	Mauritania	**UG**	Uganda
BY	Belarus	**IS**	Iceland	**MW**	Malawi	**US**	United States of America
CA	Canada	**IT**	Italy	**MX**	Mexico	**UZ**	Uzbekistan
CF	Central African Republic	**JP**	Japan	**NE**	Niger	**VN**	Viet Nam
CG	Congo	**KE**	Kenya	**NL**	Netherlands	**YU**	Yugoslavia
CH	Switzerland	**KG**	Kyrgyzstan	**NO**	Norway	**ZW**	Zimbabwe
CI	Côte d'Ivoire	**KP**	Democratic People's Republic of Korea	**NZ**	New Zealand		
CM	Cameroon			**PL**	Poland		
CN	China	**KR**	Republic of Korea	**PT**	Portugal		
CU	Cuba	**KZ**	Kazakstan	**RO**	Romania		
CZ	Czech Republic	**LC**	Saint Lucia	**RU**	Russian Federation		
DE	Germany	**LI**	Liechtenstein	**SD**	Sudan		
DK	Denmark	**LK**	Sri Lanka	**SE**	Sweden		
EE	Estonia	**LR**	Liberia	**SG**	Singapore		

RECLAIMING MEMORY FROM DELETED APPLICATIONS

Field of the Invention

The present invention is directed to the area of memory management in computer systems with segmented memory space, such as embedded devices, and specifically provides a system for safely reclaiming memory and realigning the remaining applications contiguously in non-volatile, erasable "flash"-type memory.

Background of the Invention

This application is particularly useful in embedded systems with memory constraints, but is applicable in any environment in which applications may be removed from a machine and it is desirable to compact the remaining applications stored in memory, in order to provide the maximum amount of contiguous free space for new applications.

In embedded systems, total system memory may be in the order of 1 to 4 megabytes. Ideally, half of the system memory should be devoted to read-only memory (ROM), for storing applications and data which persist between sessions. Applications can be loaded, in sequential order, until the memory limits of the system are reached.

For the purposes of this invention, reference will be made to "flash" type memory. Technically, flash is a form of ROM in that it is non-volatile, but it is also erasable, generally in 64K byte blocks, according to current technology.

Particularly in the rapidly developing area of wireless communications, it may be necessary or desirable to delete existing applications in a device before a new application can be downloaded, either to free adequate memory to accommodate the new application(s) or to avoid incompatibility with newer versions of an application.

Summary of the Invention

The present invention is directed to a system for safely removing applications from ROM, and then compacting the ROM to remove any "holes"

left by application removal. This ensures that the maximum amount of contiguous free memory is made available for new application download.

In one embodiment, the present invention is directed to an improvement in a device's non-volatile memory. This flash memory is characterized by having multiple memory segments adapted to receive and store application code and data. Preferably at least one memory segment is reserved for use during memory compaction, which is adapted to receive code and/or data copied from another memory segment of the read-only memory. A mechanism for correcting pointers within the code to reference the new memory location of the code and/or data copied from another memory segment of the read-only memory is also provided.

According to another aspect, the present invention provides a method for removing a defined body of code from a read-only memory with multiple memory segments and at least one memory segment reserved for use during compaction. In the method, if it is determined that the defined body of code overlaps into a memory segment shared with other code, the other code is copied into a memory segment reserved for use during compaction, the memory segment reserved for compaction is swapped with the memory segment shared with other code, and the memory segment containing a portion of the defined body of code is erased.

Brief Description of the Drawings

Figure 1, which comprises Figures 1A through 1D, schematically illustrates code removal and compaction in the read-only-memory (ROM) of a constrained memory device, according to the invention;

Figure 2 is a flow diagram illustrating the steps for code removal, according to the invention; and

Figure 3 is a flow diagram illustrating the steps for memory compaction following code removal, according to a preferred embodiment of the invention.

Detailed Description of the Preferred Embodiments

As schematically illustrated in Figure 1 (consisting of Figures 1A through 1D), a ROM 2 in a constrained memory device, such as a cellular

telephone, could typically comprise 32 64K memory segment blocks, generally designated by 4, to total 2 megabytes of flash memory in the device.

In the implementation of the invention for the preferred embodiment, the applications in the ROM are allocated from low memory to high memory.

As shown in Figures 1A and 1B, an application 6, shown in cross-hatch shading, spans two and a half memory segment blocks 4 in ROM 2. Code or data 8, shown in diagonal striping, from a contiguously stored application shares one memory segment with the first part of the application 6.

Only entire blocks of memory can be erased in the flash memory used in these devices, as discussed above. Therefore, if the user wants to erase the application at 6, the device's memory manager will have to save the code or data 8 from the contiguously stored application and then restore it to this location.

This is accomplished by retaining one or more spare memory blocks to use for swapping out blocks of stored code or data to retain in the device. In the preferred embodiment, the manager of the "flash" memory simply reserves the highest block(s) for the compaction process. In the example shown in Figure 1, the memory manager copies the code or data 8 up into a spare block 10 (Figure 1B). Only the code/data from the block being deleted is copied; the top of the spare block is left empty. The spare block containing the copied code 10 is then "swapped" with the original memory block at 12 (Figure 1C), and the memory blocks containing the remaining code/data of the application 6 can then be safely erased (Figure 1D). This swapping may be done in one of two ways: if a memory management unit (MMU) supporting "virtual" memory regions is available, the MMU mapping is changed so that the "new" block is using the address previously used by the block being deleted. The block being deleted is erased and its memory mapping is changed to the address being used previously by the spare block. If a MMU is not available, the contents of the "spare" block is copied on top of the original block, after it has been erased.

The steps for this process are set out in the flow diagram of Figure 2. When the user desires to delete an application from memory, the

device's memory manager locates the start and end points in ROM of the application (block 20) to determine whether both are located on a memory segment boundary (blocks 22, 24). If this is the case, it is safe for the memory manager to simply erase all memory segment blocks containing the application (block 28) and proceed to compaction which is shown in Figure 3.

Continuing with Figure 2, if the memory manager determines that an end point of the application is inside a memory segment block (block 22 or 24), it next determines whether the application is sharing a memory segment block at this point with other code (block 26). The "flash manager" keeps track of start and end addresses of each application and also knows the addresses of each memory block, so it is a simple calculation to determine where the application/data resides relative to the start/end of the memory blocks and other applications.

If no other application code/data is located, it is safe to delete the original application (block 28) and proceed to compacting the ROM.

If the memory manager determines that data or code from another application is stored in the same block with a portion of the original application for deletion (block 26), this code or data is copied into a spare memory segment block (block 28) without any of the original application code (i.e., the spare memory segment block is partially blank).

The copy to the spare memory segment block is made in such a way that the device can recover if the device is powered off in the midst of a memory deletion/compaction operation. In the preferred embodiment, flags and markers are used to indicate the state of the application during the loading, deleting and compacting functions (block 30).

The code/data from the other application in the spare memory segment block is then swapped back to the bottom of the original application to be deleted (block 34). The flags and markers added to indicate that a transfer is underway are reset (block 36) and the memory segment blocks containing the remainder of the original application are erased (block 38).

Once the application has been removed, its memory must be reclaimed by compacting or "sliding" the applications above it down into the empty space. This leaves the maximum amount of contiguous free space at the top of the ROM for loading new applications.

According to a preferred embodiment of the invention, compaction follows the same principles as for deleting an application, that is, spare memory segment blocks are used for transferring the content of partially filled memory segment blocks, and this process is illustrated in Figure 3.

Once the application has been unloaded, the memory should be compacted in order to remove any "holes". This provides the maximum contiguous free memory for downloading new applications. A preferred method for performing compaction, utilizing the principles of the present invention, is illustrated in Figure 3.

Following the application unload, the memory manager scans the memory to determine if there are any memory blocks populated with data or code above the memory blocks freed by unloading the application - that is, if unloading the application has left an empty "hole" in the recorded memory (block 40). If not, then the unloaded application was at the top of the recorded memory, and compaction is not required.

If there is recorded data/code above the freed memory, the memory manager determines whether a part or whole empty memory block is at the bottom of the freed memory (block 42).

If the freed memory is in whole memory blocks, compaction is performed simply, by copy the data blocks down the memory sequentially to the next empty memory block (block 44), and adjusting the flags/markers for the new location of each block of data or code (block 46).

Where there is only a part memory available, the memory manager copies data/code from the data block to fill the part memory block (block 48) and adjusts the flags/markers for the copied code (block 50).

In the same manner as discussed above in relation to Figure 2, the remaining data/code from the data block is copied to a spare memory block (block 52) and the flags/markers adjusted for the location of this code/data (block 54). This spare memory block is swapped to the free

memory contiguous the already compacted data (block 56), and the flags/markers of the data/code in the swapped block adjusted (block 58).

This process is performed recursively until all data has been compacted (block 40).

Although the invention has been described in association with preferred embodiments, it will be understood that it is applicable to other platforms and memory arrangements by employing modifications obvious to the person skilled in the art.

In summary there is described a method for removing code (applications and data) from read-only memory, and compacting the remaining code in memory either as an application is deleted or when there is not sufficient room to hold a new application. One or more "spare" memory segments are reserved for use during compaction. Where the code for removal shares a memory segment with other code that is not to be removed, the other code is copied to a spare memory segment, and then swapped back to its original location. The code can then be compacted to remove the "holes" left by the erased code.

CLAIMS

1. In a computing environment, a non-volatile memory having multiple memory segments adapted to receive and store application code and data, comprising:

at least one memory segment being reserved for use during memory compaction and adapted to receive only code and/or data copied from another memory segment of the non-volatile memory; and

a mechanism for correcting pointers/markers to reference a new memory location of the code and/or data copied from another memory segment of the non-volatile memory.

2. A computing environment, according to claim 1, wherein the multiple memory segments are adapted to receive data for storage from a low end to a high end and wherein said at least one memory segment is reserved at the high end.

3. A method for removing a defined body of code from a non-volatile memory having multiple memory segments, at least one memory segment being reserved for use during compaction, comprising:

scanning the defined body of code for overlap into a memory segment shared with other code;

copying the other code into a memory segment reserved for use during compaction;

swapping the other code into a memory segment reserved for use during compaction;

erasing any memory segment containing a portion of the defined body of code.

4. A method, according to claim 3 wherein the step of swapping the memory segment reserved for compaction with the memory segment shared with other code, in a system having virtual memory regions and the memory segment shared with other code is at a first address, comprising mapping the memory segment reserved for compaction to the first address.

5. A method, according to claim 3, wherein the step of swapping the memory segment reserved for compaction with the memory segment shared with other code comprises:

erasing the memory segment shared with other code; and

copying the memory segment reserved for compaction onto the erased memory segment.

6. A computer-readable memory for storing the instructions for use in the execution in a computer of any one of the methods of claims 3 through 5.

7. A computer program product comprising a computer usable medium having computer readable program code means embodied therein for causing a computer to remove a defined body of code from non-volatile memory having multiple memory segments of which at least one memory segment is reserved for use during compaction, the computer readable program code means in said computer program product comprising:

computer readable program code means for causing the computer to scan the defined body of code for overlap into a memory segment shared with other code;

computer readable program code means for causing the computer to copy the other code into a memory segment reserved for use during compaction;

computer readable program code means for causing the computer to swap the other code into a memory segment reserved for use during compaction; and

computer readable program code means for causing the computer to erase any memory segment containing a portion of the defined body of code.

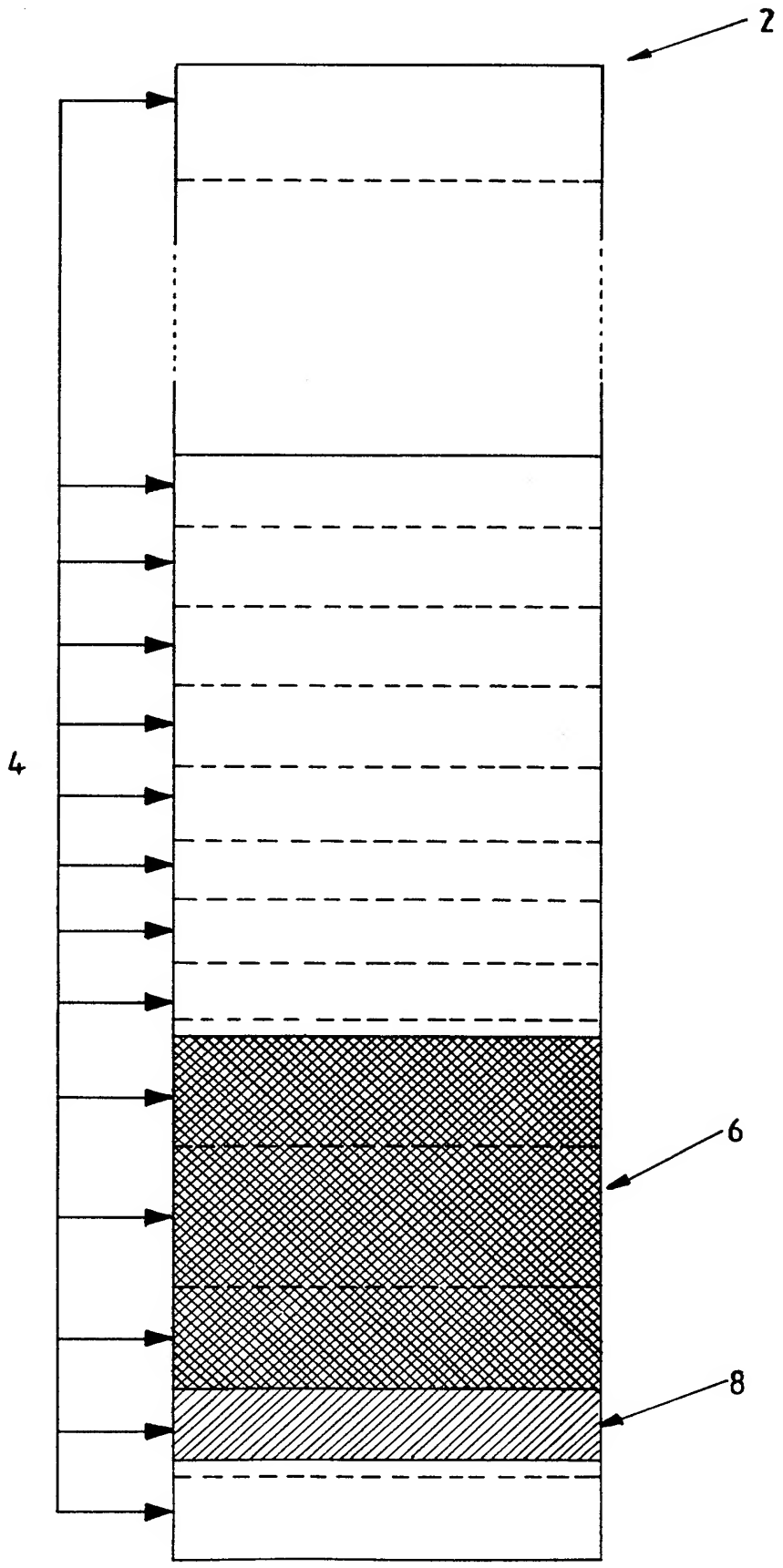


FIG. 1A

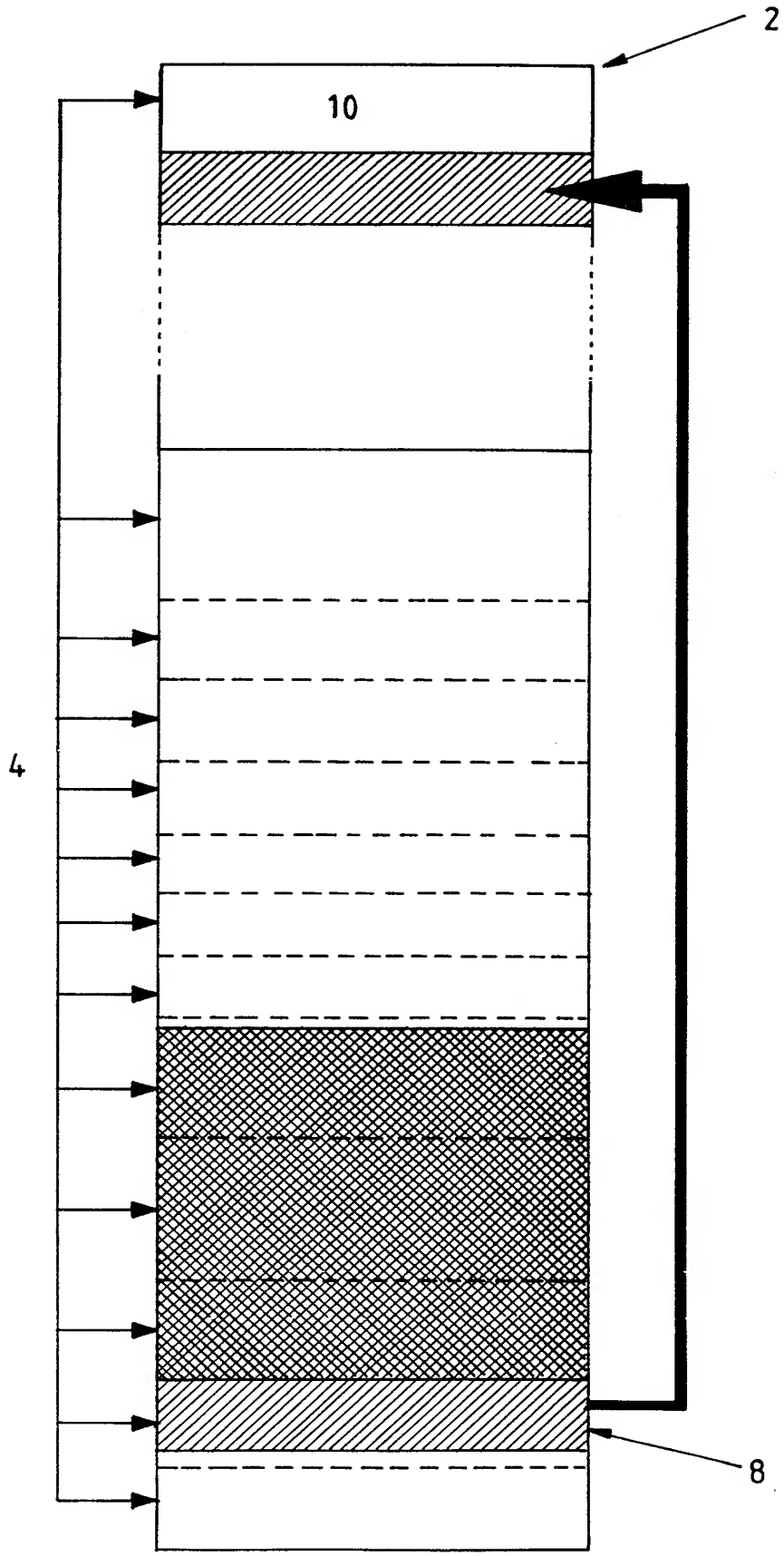


FIG. 1B

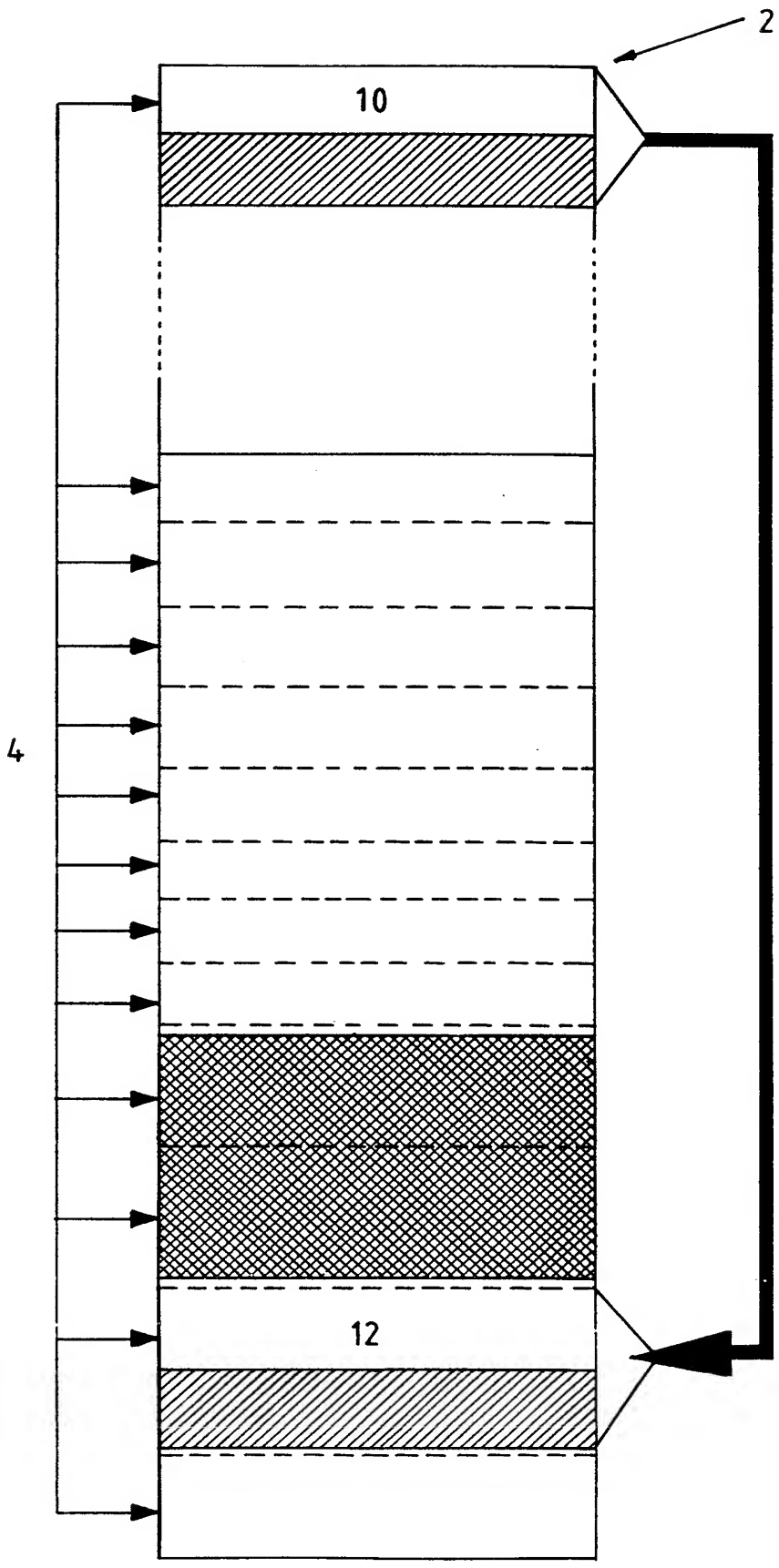


FIG. 1C

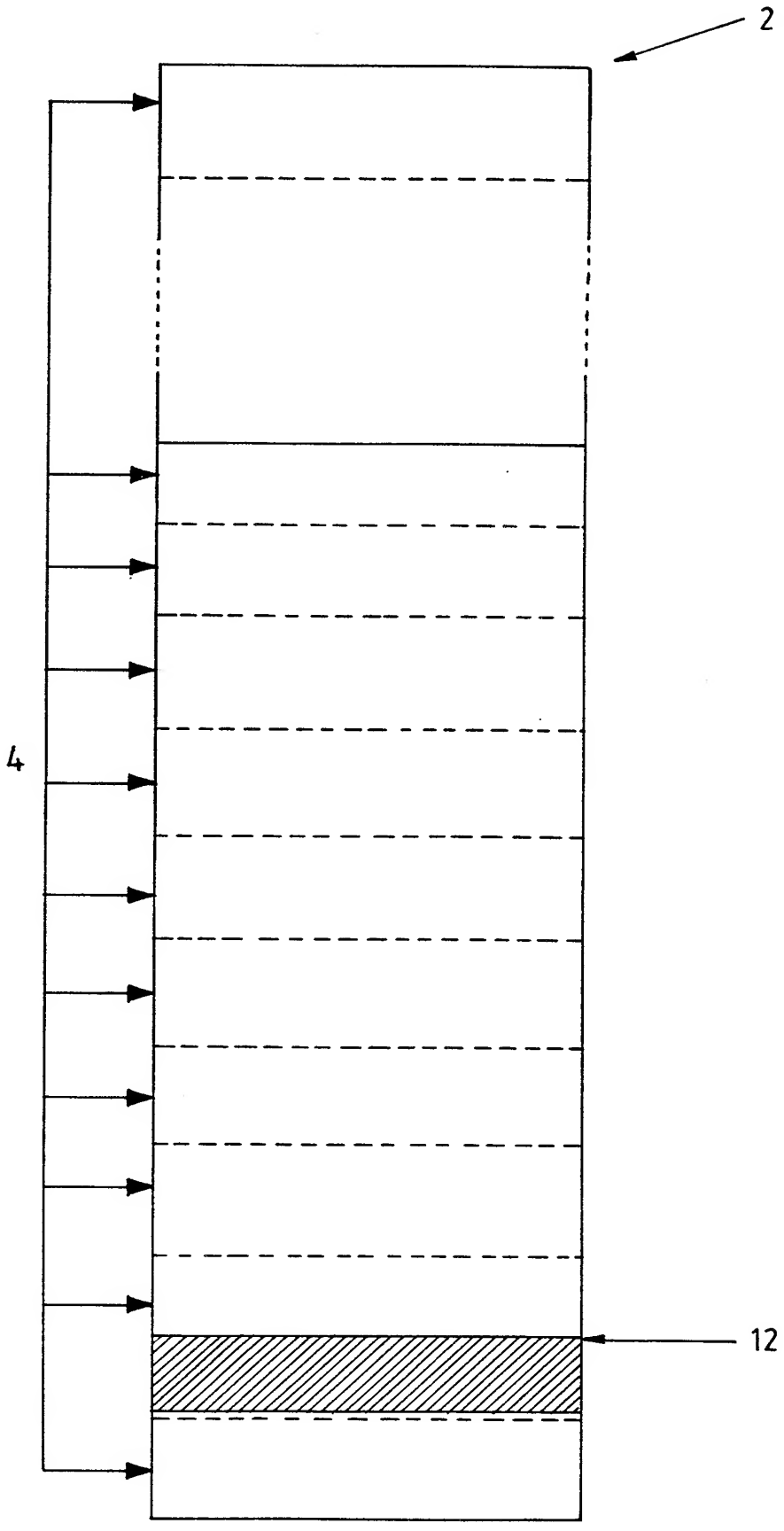
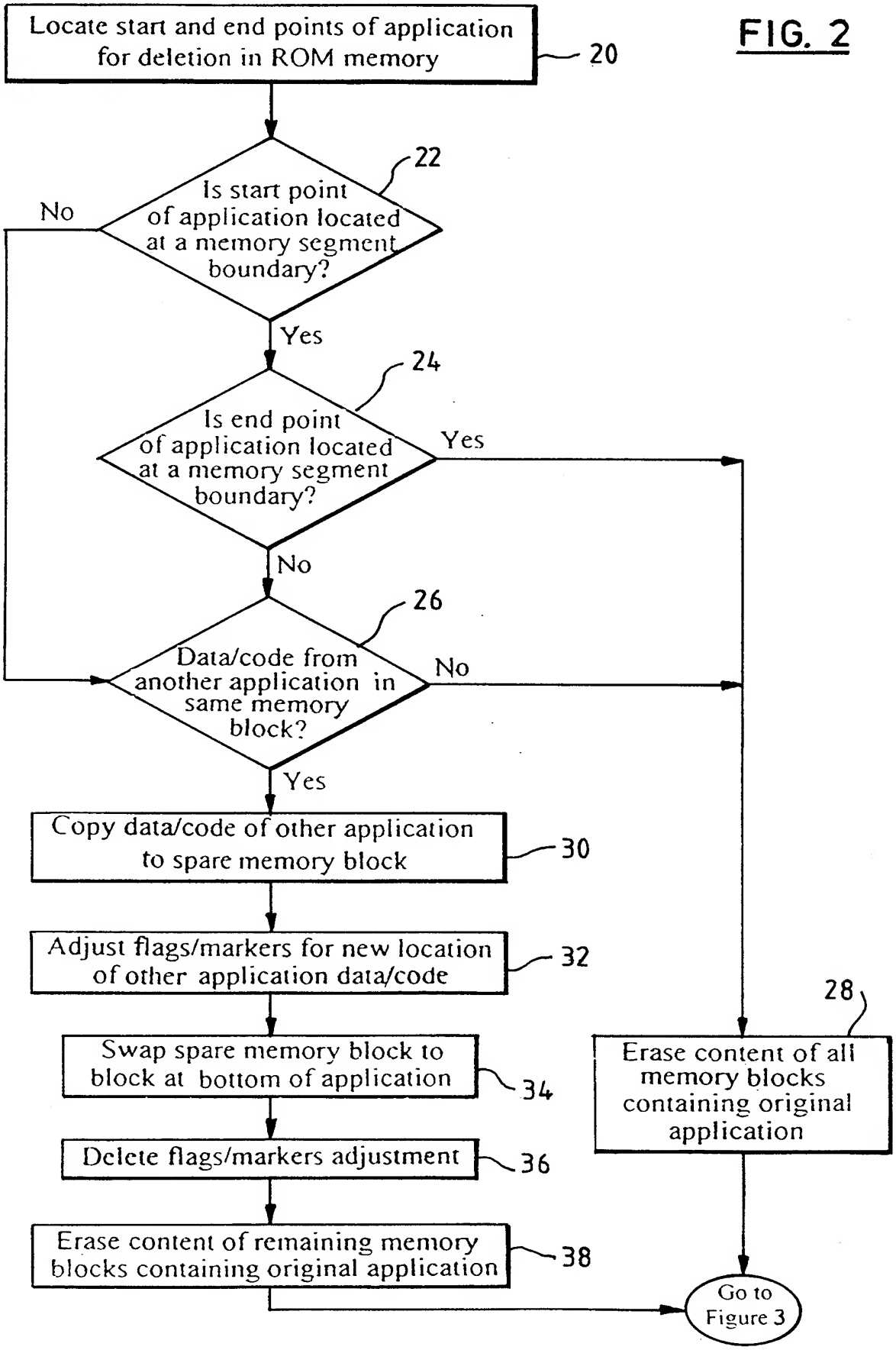


FIG. 1D

FIG. 2


Locate start and end points of application for deletion in ROM memory
20
Is start point of application located at a memory segment boundary?
22
No
Yes
Is end point of application located at a memory segment boundary?
24
Yes
No
Data/code from another application in same memory block?
26
No
Yes
Copy data/code of other application to spare memory block
30
Adjust flags/markers for new location of other application data/code
32
Swap spare memory block to block at bottom of application
34
Delete flags/markers adjustment
36
Erase content of remaining memory blocks containing original application
38
Erase content of all memory blocks containing original application
28
Go to Figure 3

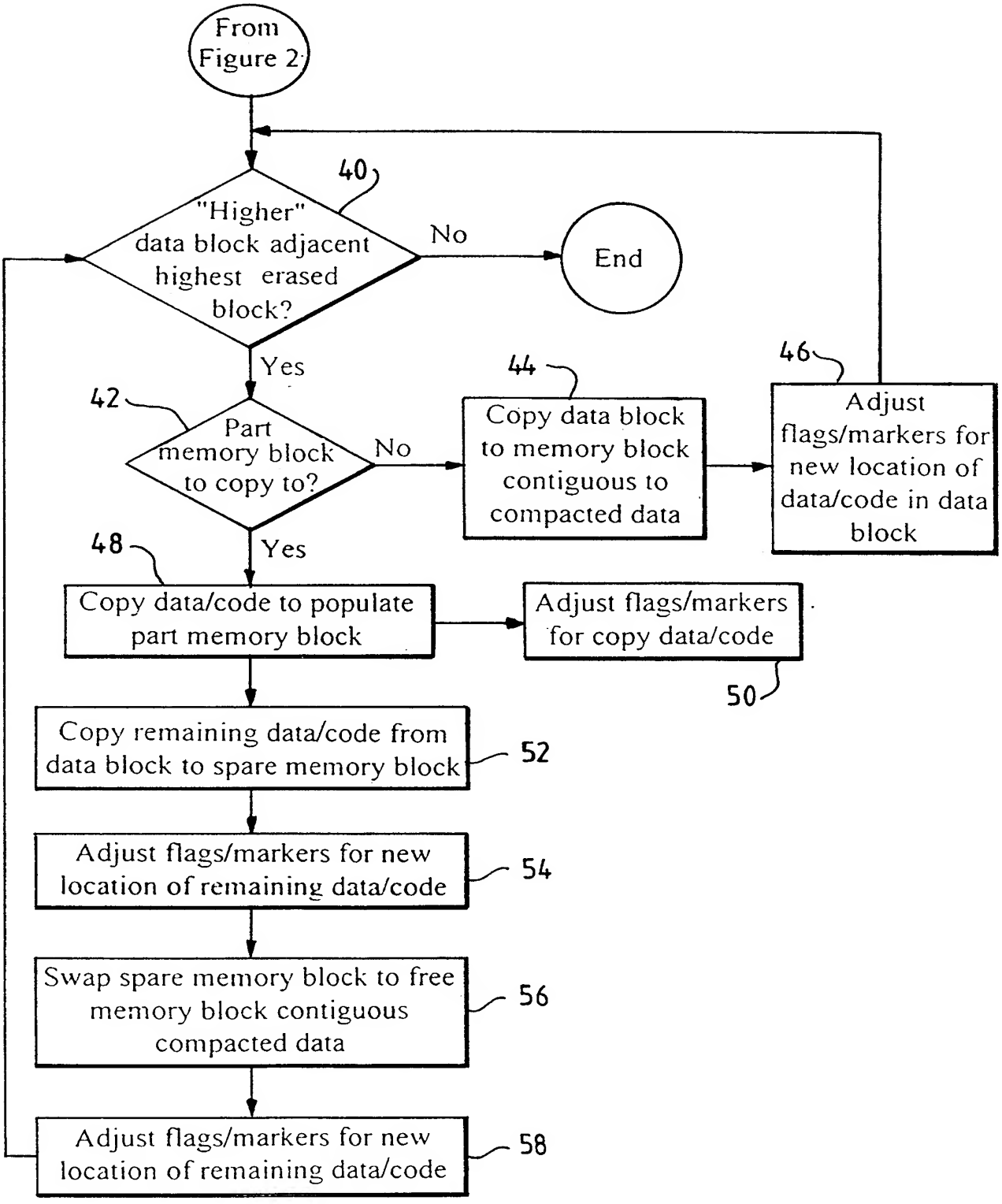

From Figure 2
40
"Higher" data block adjacent highest erased block?
No
End
Yes
42
Part memory block to copy to?
No
44
Copy data block to memory block contiguous to compacted data
46
Adjust flags/markers for new location of data/code in data block
Yes
48
Copy data/code to populate part memory block
Adjust flags/markers for copy data/code
50
Copy remaining data/code from data block to spare memory block
52
Adjust flags/markers for new location of remaining data/code
54
Swap spare memory block to free memory block contiguous compacted data
56
Adjust flags/markers for new location of remaining data/code
58

FIG. 3

INTERNATIONAL SEARCH REPORT

Inter nal Application No
PCT/GB 00/01063

A. CLASSIFICATION OF SUBJECT MATTER
IPC 7 G06F12/02

According to International Patent Classification (IPC) or to both national classification and IPC

B. FIELDS SEARCHED

Minimum documentation searched (classification system followed by classification symbols)
IPC 7 G06F

Documentation searched other than minimum documentation to the extent that such documents are included in the fields searched

Electronic data base consulted during the international search (name of data base and, where practical, search terms used)

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category °	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
X	WO 94 20906 A (SYSTEMS LTD M ;SYSTEMS INC M (US)) 15 September 1994 (1994-09-15) page 5, line 23 -page 7, line 29; figures 1,2,4 page 9, line 23 -page 10, line 11; figure 7	1-4,6,7
X	EP 0 745 939 A (LUCENT TECHNOLOGIES INC) 4 December 1996 (1996-12-04) page 3, line 52 -page 4, line 12 page 6, line 19 - line 28; figure 2	1-3,5-7

☐ Further documents are listed in the continuation of box C.

☒ Patent family members are listed in annex.

° Special categories of cited documents :

"A" document defining the general state of the art which is not considered to be of particular relevance

"E" earlier document but published on or after the international filing date

"L" document which may throw doubts on priority claim(s) or which is cited to establish the publication date of another citation or other special reason (as specified)

"O" document referring to an oral disclosure, use, exhibition or other means

"P" document published prior to the international filing date but later than the priority date claimed

"T" later document published after the international filing date or priority date and not in conflict with the application but cited to understand the principle or theory underlying the invention

"X" document of particular relevance; the claimed invention cannot be considered novel or cannot be considered to involve an inventive step when the document is taken alone

"Y" document of particular relevance; the claimed invention cannot be considered to involve an inventive step when the document is combined with one or more other such documents, such combination being obvious to a person skilled in the art.

"&" document member of the same patent family

Date of the actual completion of the international search	Date of mailing of the international search report
6 June 2000	14/06/2000

Name and mailing address of the ISA	Authorized officer
European Patent Office, P.B. 5818 Patentlaan 2 NL – 2280 HV Rijswijk Tel. (+31–70) 340–2040, Tx. 31 651 epo nl, Fax: (+31–70) 340–3016	Ledrut, P

1

Form PCT/ISA/210 (second sheet) (July 1992)

INTERNATIONAL SEARCH REPORT

Information on patent family members

Inter · nal Application No
PCT/GB 00/01063

Patent document cited in search report	Publication date	Patent family member(s)	Publication date
WO 9420906 A	15-09-1994	US 5404485 A	04-04-1995
		AU 6269994 A	26-09-1994
		CN 1098526 A	08-02-1995
		DE 69414556 D	17-12-1998
		DE 69414556 T	06-05-1999
		EP 0688450 A	27-12-1995
		FI 954235 A	08-11-1995
		IL 108766 A	05-12-1996
		JP 8510072 T	22-10-1996
		ZA 9401446 A	26-09-1994
EP 0745939 A	04-12-1996	JP 9152983 A	10-06-1997